USFD: Preliminary Exploration of Features and Classifiers for the TempEval-2007 Tasks

Mark Hepple
Dept of Computer Science
University of Sheffield
Regent Court
211 Portobello Street
Sheffield S1 4DP, UK
hepple@dcs.shef.ac.uk

Andrea Setzer
Dept of Computer Science
University of Sheffield
Regent Court
211 Portobello Street
Sheffield S1 4DP, UK
andrea@dcs.shef.ac.uk

Rob Gaizauskas
Dept of Computer Science
University of Sheffield
Regent Court
211 Portobello Street
Sheffield S1 4DP, UK
robertg@dcs.shef.ac.uk

Abstract

We describe the Sheffield system used in TempEval-2007. Our system takes a machine-learning (ML) based approach, treating temporal relation assignment as a simple classification task and using features easily derived from the TempEval data, i.e. which do not require 'deeper' NLP analysis. We aimed to explore three questions: (1) How well would a 'lite' approach of this kind perform? (2) Which features contribute positively to system performance? (3) Which ML algorithm is better suited for the TempEval tasks? We used the Weka ML workbench to facilitate experimenting with different ML algorithms. The paper describes our system and supplies preliminary answers to the above questions.

1 Introduction

The Sheffield team were involved in TempEval as co-proposers/co-organisers of the task.[1] For our participation in the task, we decided to pursue an ML-based approach, the benefits of which have been explored elsewhere (Boguraev and Ando, 2005; Mani et al., 2006). For the TempEval tasks, this is easily done by treating the assignment of temporal relation types as a simple classification task, using readily available information for the instance features. More specifically, the features used were ones provided as attributes in the TempEval data annotation for the events/times being related, plus some additional features that could be straightforwardly computed from documents, i.e. without the use of more heavily 'engineered' NLP components. The aims of this work were three-fold. First, we wanted to see whether a 'lite' approach of this kind could yield reasonable performance, before pursuing possibilities that relied on using 'deeper' NLP analysis methods. Secondly, we were interested to see which of the features considered would contribute positively to system performance. Thirdly, rather than selecting a single ML approach (e.g. one of those currently in vogue within NLP), we wanted to look across ML algorithms to see if any approach was better suited to the TempEval tasks than any other, and consequently we used the Weka workbench (Witten and Frank, 2005) in our ML experiments.

In what follows, we will first describe how our system was constructed, before going on to discuss our main observations around the key aims mentioned above. For example, in regard to our 'lite' approach, we would observe (c.f. the results reported in the Task Description paper) that although some other systems scored more highly, the score differences were relatively small. Regarding features, we found for example that the system performed better for Task A when, surprisingly, the `tense` attribute of EVENTs was *excluded*. Regarding ML algorithms, we found not only that there was substantial variation between the effectiveness of different algorithms for assigning relations (as one might expect), but also that there was considerable differences in the relative effectiveness of algorithms *across* tasks,

[1] We maintained a strict separation between persons assisting in annotation of the test corpus and those involved in system development.

i.e. so that an algorithm performing well on one task (compared to the alternatives), might perform rather poorly on another task. The paper closes with some comments about future research directions.

2 System Description

The TempEval training and test data is marked up to identify all event and time expressions occurring within documents, and also to record the TLINK relations that are relevant for each task (except that TLINK relation types are absent in the test data). These annotations provide additional information about these entities in the form of XML attributes, e.g. for EVENT annotations we find attributes such as tense, aspect, part-of-speech and so on.

Our system consists of a suite of Perl scripts that create the input files required for Weka, and handle its output. These include firstly an 'extraction' script, which extracts information about EVENT, TIMEXs and TLINKs from the data files, and secondly a 'feature selection/reformatting' script, which allows the information that is to be supplied to Weka to be selected, and recasts it into the format that Weka requires for its training/test files. A final script takes Weka's output over the test files and connects it back to the original test documents to produce the final output files required for scoring.

The information that the first extraction script extracts for each EVENT, TIMEX and TLINK largely corresponds to attributes/values associated with the annotations of these items in the initial data files (although not all such attributes are of use for machine learning purposes). In addition, the script determines for each EVENT expression whether it is one deemed relevant by the Event Target List (ETL) for Tasks A and B. This script also maps EVENTs and TIMEXs into sequential order – intra-sentential order for task A and inter-sentential order for task C. This information can be used to compute various 'order' features, such as:

`event-first`: do a related EVENT and TIMEX (for Task A) appear with the EVENT before or after the TIMEX?

`adjacent`: do a related EVENT and TIMEX (again for Task A) appear adjacently in the sequence of temporal entities or not? (Note that this allows an EVENT and TIMEX to be adjacent if there tokens that intervene, but not any other temporal entities.)

`event-between`: for a related EVENT/TIMEX pair, do any other events appear between them?

`timex-between`: for a related EVENT/TIMEX pair, do any other timexes appear between them?

		Task		
Type	Attribute	A	B	C
EVENT	aspect	✓	✓	✓
EVENT	polarity	✓	✓	×
EVENT	POS	✓	✓	✓
EVENT	stem	✓	×	×
EVENT	string	×	×	×
EVENT	class	×	✓	✓
EVENT	tense	×	✓	✓
ORDER	adjacent	✓	N/A	N/A
ORDER	event-first	✓	N/A	N/A
ORDER	event-between	×	N/A	N/A
ORDER	timex-between	×	N/A	N/A
TIMEX3	mod	✓	×	N/A
TIMEX3	type	✓	×	N/A
TLINK	reltype	✓	✓	✓

Table 1: Features

Table 1 lists all the features that we tried using for any of the three tasks. Aside from the ORDER features (as designated in the leftmost column), which were computed as just described, and the EVENT `string` feature (which is the literal tagged expression from the text), all other features correspond to annotation attributes. Note that the TLINK `reltype` is extracted from the training data to provide the target attribute for training (a dummy value is provided for this in test data).

The output of the extraction script is converted to a format suitable for use by Weka by a second script. This script also allows a manual selection to be made as to the features that are included. For each of the three tasks, a rough-and-ready process was followed to find a 'good' set of features for use with that task, which proceeded as follows. Firstly, the maximal set of features considered for the task was tried with a few ML algorithms in Weka (using a 10-fold cross-validation over the training data) to find one that seemed to work quite well for the task. Then using only that algorithm, we checked whether the `string` feature could be dropped (since this feature's value set was always of quite high cardinality), i.e. if its omission improved performance, which for all three tasks was the case. Next, we tried dropping each of the remaining features in turn, to identify those whose exclusion improved performance, and then for those features so identified, tried dropping them in combination to arrive at a final 'optimal' feature set. Table 1 shows for each of the tasks which of the features were considered for inclusion (those marked N/A were *not*), and which of these remained in the final optimal feature set (✓).

Having determined the set of features for use with each task, we tried out a range of ML algorithms (again with a 10-fold cross-validation over the training data), to arrive at the final feature-set/ML algorithm combination that was used for the task in the competitive evaluation. This was trained over the entire training data and applied to the test data to produce the final submitted results.

3 Discussion

Looking to Table 1, and the features that were considered for each task and then included in the final set, various observations can be made. First, note that the `string` feature was omitted for all tasks, which is perhaps not surprising, since its values will be sparsely distributed, so that there will be very few training instances for most of its individual values. However, the `stem` feature was found to be useful for Task A, which can be interpreted as evidence for a 'lexical effect' on local event-timex relations, e.g. perhaps with different verbs displaying different trends in how they relate to timexes. No corresponding effects were observed for Tasks B and C.

The use of ORDER features for Task A *was* found to be useful – specifically the features indicating whether the event or timex appeared linearly first in the sentence and whether the two were adjacent or not. The more elaborate ORDER features, addressing more specific cases of what might intervene between the related timex and event expression, were not found to be helpful.

Perhaps the most striking observation to be made regarding the table is that it was found beneficial to exclude the feature `tense` for Task A, whilst the feature `aspect` was retained. We have no explanation to offer for this result. Likewise, the event `class` feature, which distinguishes e.g. perception vs. reporting vs. aspectual etc verbs, was excluded for Task A, although it was retained for Task B.

	Task		
Algorithm	A	B	C
`baseline`	49.8	62.1	42.0
`lazy.KStar`	**58.2**	76.7	54.0
`rules.DecisionTable`	53.3	**79.0**	52.9
`functions.SMO` (svm)	55.1	78.1	**55.5**
`rules.JRip`	50.7	78.6	53.4
`bayes.NaiveBayes`	56.3	76.2	50.7

Table 2: Comparing different algorithms (%-acc. scores, from cross-validation over training data)

In regard to the use of different ML algorithms for the classification tasks addressed in TempEval, we observed considerable variation between algorithms as to their performance, and this was not unexpected. However, given the seemingly high similarity of the three tasks, we were rather more surprised to see that there was considerable variation between the performance of algorithms *across* tasks, i.e. so that an algorithm performing well on one task (compared to the alternatives), might perform rather poorly on another task. This is illustrated by the results in Table 2 for a selected subset of the algorithms considered, which shows %-accuracy scores that were computed by cross-validation over the training data, using the feature set chosen as 'optimal' for each task.[2] The algorithm names in the left-hand column are the ones used in WEKA (of which `functions.SMO` is the WEKA implementation of support-vector machines or SVM). The first row of results give a 'baseline' for performance, corresponding to the assignment of the most common label for the task. (These were produced using WEKA's `rules.ZeroR` algorithm, which does exactly that.)

The best results observed for each task are shown in bold in the table. These best performing algorithms were used for the corresponding tasks in the competition. Observe that the `lazy.KStar`

[2]These scores are computed under the 'strict' requirement that key and response labels should be identical. The TempEval competition also uses a 'relaxed' metric which gives partial credit when one (or both) label is disjunctive and there is a partial match, e.g. between labels AFTER and OVERLAP-OR-AFTER. See (Verhagen et al., 2007) for details.

	Task A		Task B		Task C	
	F_S	F_R	F_S	F_R	F_S	F_R
USFD	0.59	0.60	0.73	0.74	0.54	0.59
ave.	0.56	0.59	0.74	0.75	0.51	0.60
max.	0.62	0.64	0.80	0.81	0.55	0.66

Table 3: Competition task scores for Sheffield system (USFD), plus average/max scores across all competing systems

method, which gives the best performance for Task A, gives a rather 'middling' performance for Task B. Similarly, the SVM method that gives the best results for Task C falls quite a way below the performance of `KStar` on Task A. A more extreme case is seen with the results for `rules.JRip` (Weka's implementation of the RIPPER algorithm), whose score for Task B is close to that of the best-performing system, but which scores only slightly above baseline on Task A.

The competition scores for our system are given in Table 3, shown as (harmonic) F-measures under both strict (F_S) and relaxed (F_R) metrics (see footnote 2). The table also shows the average score for each task/metric across all systems taking part in the competition, as well as the maximum score returned by any system. See (Verhagen et al., 2007) for a full tabulation of results for all systems.[3]

4 Future Directions

SIGNALs and SLINKs are possible candidates as additional features – signals obviously so, whereas the benefits of exploiting subordination information are less clear. Our initial exploratory efforts in this direction involved pulling information regarding SIGNALs and SLINKs across from TimeBank[4] (Pustejovsky et al., 2003) so as to make this available for use with the TempEval tasks, in the hope that this would allow us to determine if this information would be useful without first facing the cost of developing SIGNAL and SLINK recognisers. Regarding SIGNALs, however, we ran into the problem that there are many TLINKs in the TempEval data for which no corresponding TLINK appears in TimeBank, and hence for which SIGNAL information could not be imported. We were unable to progress this work sufficiently in the time available for there to be any useful results to report here.

[3]The TempEval test data identifies precisely the temporal entity pairs to which a relation label must be assigned. When a fixed set of items is classified, the scores for precision, recall and F-measure will be identical, being the same as the score for simple accuracy. However, not all the participating systems follow this pattern of assigning labels to 'all and only' the entity pairs identified in the test data, i.e. some systems decide which entity pairs to label, as well as which label to assign. Accordingly, the performance results given in (Verhagen et al., 2007) are reported using metrics of precision, recall and F-measure.

[4]This was possible because both the trial and training data were derived from TimeBank.

5 Conclusion

We have explored using a ML-based approach to the TempEval tasks, which does not rely on the use of deeper NLP-analysis components. We observe that although some other systems in the competition have produced higher scores for the tasks, the score differences are relatively small. In the course of this work, we have made some interesting observations regarding the performance variability of different ML algorithms when applied to the diffent TempEval tasks, and regarding the features that contribute to the system's performance.

References

B. Boguraev and R. Kubota Ando. 2005. TimeML-Compliant Text Analysis for Temporal Reasoning. In *Proceedings of IJCAI-05*, pages 997–1003.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine Learning of Temporal Relations. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 753–760, Morristown, NJ, USA. Association for Computational Linguistics.

J. Pustejovsky, D. Day, L. Ferro, R. Gaizauskas, P. Hanks, M. Lazo, D. Radev, R. Saurí, A. See, A. Setzer, and B. Sundheim. 2003. The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656, Lancaster, March.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of SemEval-2007: 4th International Workshop on Semantic Evaluations.*

I.H. Witten and E. Frank, editors. 2005. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, San Francisco, second edition.